Now, note the following. The value of the Production Array at a value of Shift equal to 7 is 7 and it is the last member of the array (Figure 130). Watch what happens when Shift is equal to 8 and what is shown in the contents of Production[8], because there is not really a Production[8] array member. After we try to store the value in Production[8] with the Move instruction the system will fault.
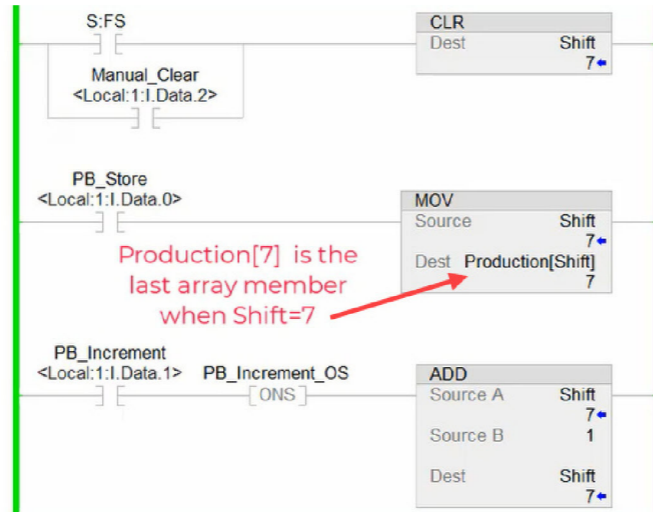
Now, if we add 1 to Shift (Figure 131), and Shift becomes 8, the value of Production[8] shows ?? (question marks), because the controller is showing us that is does not really have a value to show because it is actually an invalid location, Production[8] does not exist. However, a fault is not yet generated because we have not used the invalid Array address yet. The moment that we try to store the value of 8 into Production[8], the system will crash.
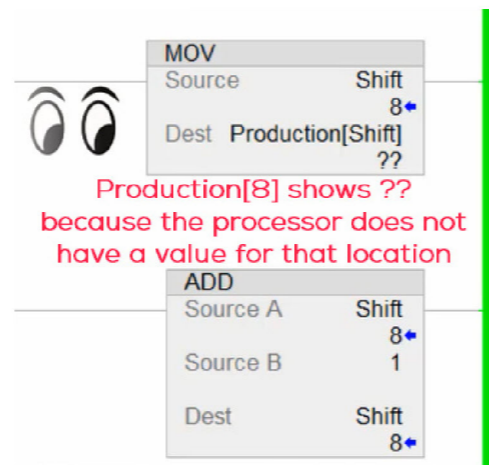
If the PB_Store pushbutton is pressed (Figure 132), the controller will immediately generates a fault. As we will see later in the program when we are troubleshooting controller's problems, the controller fault indicator starts flashing and the Mode Status states that a fault has occurred.
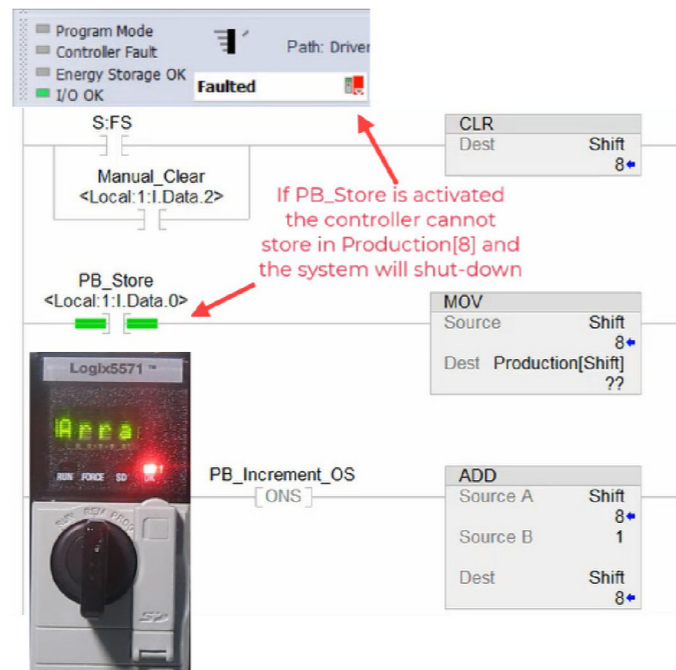
If we open the controller properties (Figure 133), it will indicate in the Major Fault tab that a fault has occurred because of an "Array Subscript Too Large".
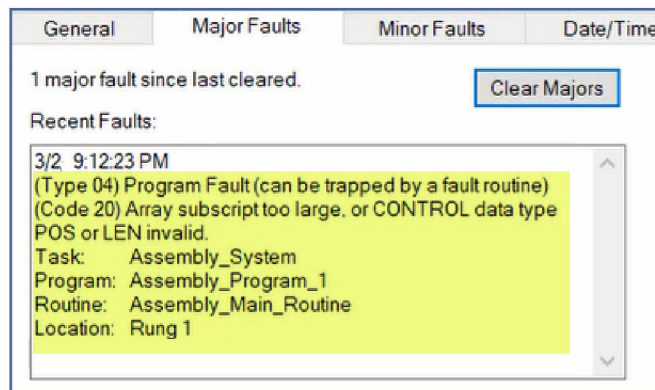

Figure 130. Storing last member of the array [7].

Figure 131. Array content display of a non-existent member.

Figure 133. Fault is recorded in the Controller's properties

Figure 132. Fault when boundary is passed.

If we clear the faults, and place the controller back into the Run Mode, the controller will run again, however the same problem will occur again when the value of Shift is again eight.

This "bug" in the code can be easily fixed by adding a rung which compares if the value of Shift is equal to 8 (Figure 134), and if so, to clear the value of Shift back to 0. In this manner the Production Array will never be addressed by a Shift tag greater than 7. In our program the bug-fixing rung will reset the Shift tag back to 0 when the Shift goes from 7 to 8.
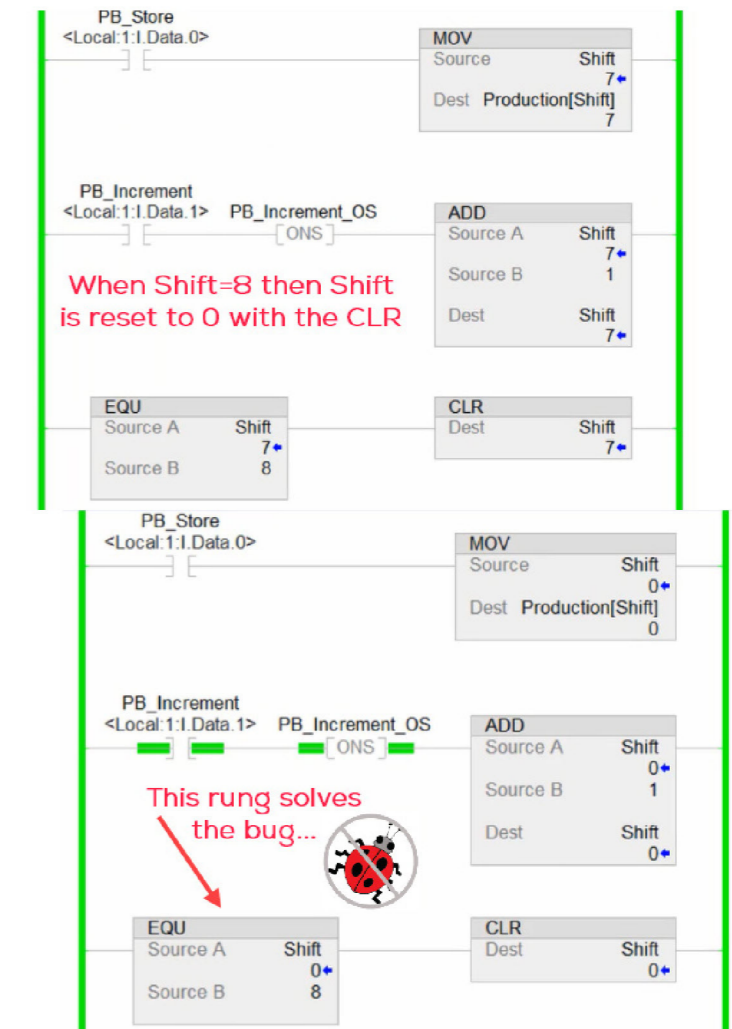
Other Array Indirect Addressing

Arrays can also be indirect addressed by an expression, such as Shift *2, or [(Shift*2)/Step] as shown in Figure 135. As you can see, using expressions can complicate the indirect addressing of the Array and we need to be prepared to troubleshoot any type of addressing.
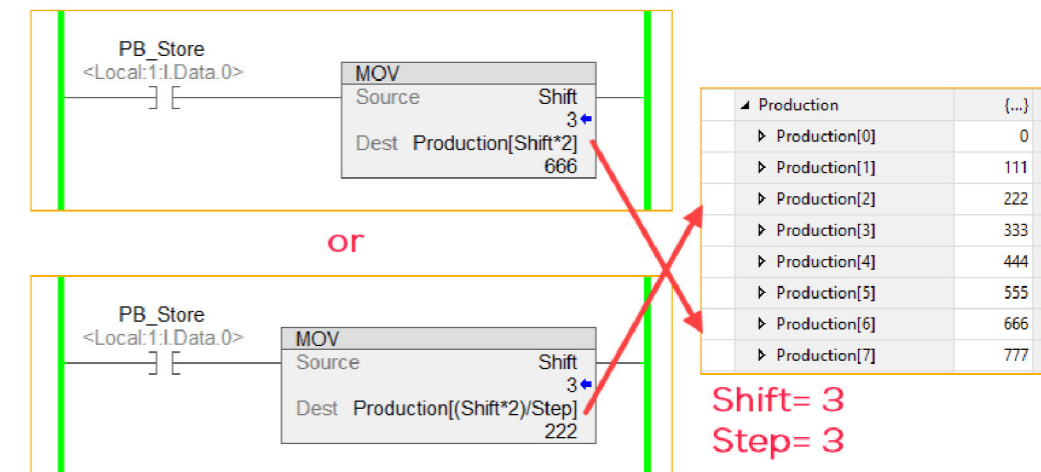

Figure 134. Fixing the going beyond the array boundary.

Figure 135. Indirect addressing using expressions.